Design Philosophy of a Networking-Oriented Data-Driven Processor: CUE

Hiroaki NISHIKAWA (a), Member

SUMMARY  To realize a secure networking infrastructure, the author is carrying out CUE (Coordinating Users’ requirements and Engineering constraints) project with a network carrier and a VLSI manufacturer. Since CUE-series data-driven processors developed in the project were specifically designed to be an embedded programmable component as well as a multi-processor element, particular design considerations were taken to achieve real-time multiprocessor capabilities essentially needed in multimedia communication environment. A novel data-driven paradigm is first introduced with special emphasis on VLSI-oriented parallel processing architectures. Data-driven protocol handlings on CUE-p and CUE-v1 are then discussed for their real-time multiprocessor capability without any runtime overheads. The emulation facility RESCUE (Real-time Execution System for CUE-series data-driven processors) was also built to develop scalable chip multi-processors in self-evolutional manner. Based on emulation results, the latest version named CUE-v2 was realized as a hybrid processor enabling simultaneous processing of data-driven and control-driven threads to achieve higher performance for inline processing and to avoid any bottlenecks in sequential parts of real-time programs frequently encountered in actual time-sensitive applications. Effectiveness of the data-driven chip multi-processor architecture will finally be addressed for lower power consumption and scalability to realize future VLSI processors in the sub-100 nm era.

key words: data-driven, multiprocessing, real-time, VLSI

1. Introduction

What should future information infrastructure be? At present, it is not so easy to get an exact answer, and at the same time, it is one of the most important issues to research and develop information communication environment. Undoubtedly, users’ requirements and engineering constraints will give a great influence onto the future infrastructure. The author therefore is carrying out a project named CUE [1]. The CUE project was started to take up real-time multiprocessing capability as a crucial users’ requirement and VLSI-oriented processor architecture as a practical engineering constraint. After developing protocol handling prototypes [2] and their emulation studies, the latest version data-driven processor CUE-v2 [3], [4] was developed. The CUE-v2 was realized as a hybrid processor enabling simultaneous processing of data-driven and control-driven threads to achieve higher performance for inline processing and to avoid any bottlenecks in sequential parts of real-time programs frequently encountered in actual time-sensitive applications.

This paper first introduces novel data-driven paradigm to show scalable implementation of VLSI-oriented processors. Data-driven parallel implementation of protocol handling is then proposed to demonstrate efficient real-time execution without any supervisory controls. Thirdly, the latest version networking-oriented data-driven processor CUE-v2 based on emulation results of RESCUE [5] is briefly introduced. Finally, effectiveness of the chip multi-processor architecture of the CUE-series data-driven processors will be discussed for lower power consumption and scalability to realize future VLSI processors in the sub-100 nm era.

2. Data-Driven Paradigm for VLSI-Oriented Parallel Processor

As shown in Fig. 1, processor cores in the data-driven chip multi-processor CUE was derived from Q-series data-driven processor project, which was inaugurated in fiscal year 1982–1983 by a feasibility study on possible application of a data-driven processing concept for realization of a high throughput embedded functional-device on one chip. This section introduces effective stream processing based on dynamic data-driven execution scheme that gives rise to a unique “flow-thru” processing scheme in which all processing, data-transfer and storage functions are carried out highly-parallel by a packetized data-flow through the elastic pipeline stages with completely distributed control.

2.1 Stream Processing Based on Dynamic Data-Driven Execution Scheme

A highly efficient multimedia stream processing scheme becomes essential to develop an information system environment as infrastructure fully utilizing the communication capabilities of so called broadband networks. Effective processing of multiple multimedia streams on the network, which are characterized by magnitude as well as their variety in time constraints, firstly requires an extensive utilization of parallel processing scheme without any bottlenecks. In order to realize the stream processing scheme, the author has been studying on data-driven chip multi-processors named CUE as shown in Fig. 1.

Generally, the data-flow or data-driven processors are classified two broader categories, static or dynamic [6] by the mode of holding the active tokens or packets in the system. The author focused on the fact that dynamic data-driven principle can naturally represent parallel process in...
concurrency, pipelining and multi-processing. As a result, the author introduced the concept of generation in addition to the so-called color as tags of tokens and proposed a Diagrammatical Data-Driven Language (D^3L) [7] in the feasibility study of the development in Fig. 1. In the D^3L program, data structure “stream” is defined by tokens with consecutive generation. Through experimental evaluations for D^3L program, data structure “stream” can be executed highly parallel as far as a system does not have any bottlenecks in its data flow paths.

The stream processing emphasizes the receiving capability of the input stream without obstructing it rather than attempting to speed up individual processing as have done in the past. Accordingly, in the system design of the first version prototype data-driven processor Q-p in Fig. 1, it was first decided that the processor has to be realized by extensive pipeline processing scheme. With the pipeline scheme, receiving capability of the data volume can be enlarged by subdividing stages. In order for the pipeline processing scheme to ideally operate, the followings must be satisfied simultaneously:

- Pipeline must not be starved, i.e. data flow must be sufficiently supplied to pipeline input.
- Data in the pipeline must not be flushed, and data flow volume must be maintained.
- Momentary fluctuations of the input to the pipeline must not degrade the pipeline throughput.
- Momentary fluctuations in data-flow-rate in the pipeline must not obstruct input to the pipeline.

As was mentioned earlier, by employing dynamic data-driven scheme, streams are realized in a parallel executable format as tagged token queues which can be effective in avoiding starvation. Also, as long as sufficient data flow is supplied, processing delay is tolerable in the data-driven scheme which is an excellent feature that maximizes the pipelining efficiency. Furthermore, by the data-driven firing rule, processing is always in a forward direction and never requiring flushing.

Thus, the key to realization of an effective stream processing based on the dynamic data-driven execution scheme lies in how momentum data flow fluctuations to and in the pipeline is absorbed.

2.2 Study of “Flow-Thru” Processing Scheme

As it is relevant from the firing rule in the data-driven scheme, functional processes or memory accesses which involve history-sensitive processes proceed by sending and receiving input/output tokens in conjunction with data-driven firing control mechanism. In general, data-driven execution of parallel processes has data fluctuations due to partial ordering among processes. Therefore, in order to maintain an average data flow rate, there needs to be some buffering mechanism between mutually connected functions. Furthermore, in order to maximize the advantage of the pipeline processing, this buffering mechanism must be realized so as to avoid bottlenecks in processing functions in distribution of load and function. Thus, the effective buffering mechanism is essential to realize efficient pipeline scheme as well as the dynamic data-driven execution itself.

The author, in order to satisfy these requirements, focused on the following issues:

- Realizing a comprehensive autonomously controlled elastic pipeline structure for the entire data path necessary for data-driven execution which can then effectively buffer fluctuations in data flow.

Further, in order to locally and autonomously control
these buffering effects,

- Enabling functions to the branch/joint/routing mechanisms so that the data can autonomously travel the appointed path.

With these fundamental concepts in mind, the Q-p was developed [8]. The Q-p's data processing and transfer functions are basically composed of a queue carrying packets representing the tokens. Therefore, all data-transfer, processing and storage functions in the Q-p are carried out data packets flowing through the elastic pipeline as shown in Fig. 2. The underlying basic architectural design principles presented here are collectively called the “flow-thru” processing scheme. With this totally pipelined configuration, neither clock signals nor passive busses were employed in the Q-p. It is also noted that no central control resides in this processor. Furthermore, multi-processor configuration to achieve higher performance is realized by directly interconnecting processing elements as shown in Fig. 2.

These features have contributed to fully utilize the advantage of the VLSI fabrication progress. In fact, the single chip data-driven processor Q-x was developed after experimental studies by multi chip version Q-v1. Furthermore, the first chip multi-processor version of our data-driven processor, which was implemented by super-integrating 4 processing elements and achieved sufficient throughput to process high definition video signals. Presently, the data-driven chip multi-processor CUE-v2 as the latest version was realized in self-evolutional manner through developing CUE-p and CUE-v1 as shown in Fig. 1.

3. Data-Driven Protocol Handling

As shown in Fig. 1, the CUE project was inaugurated in 1995 by feasibility study on possible data-driven implementation of real-time multiprocessing essentially needed in protocol handling for multi-media networking environment. In order to keep the maximum throughput in our data-driven processor discussed in Sect. 2, the basic design target for real-time multiprocessing scheme was chosen so as to alleviate any runtime overheads and to achieve real-timeness in the protocol handling without any supervisory controls. This section first addresses needs of real-time multiprocessing in the networking environment and then shows ineffectiveness of conventional implementations of protocol handling. The data-driven implementations of protocol handling on CUE-p and CUE-v1 will be finally introduced to demonstrate their effectiveness.

3.1 Needs of Real-Time Multi-Processing

In the recent networking environment [9], such many types of information as data, audio stream, video stream, etc. are handled so concurrently and effectively in each layer that multi-processing is needed. A typical example of multimedia networking environment is CDN (Contents Delivery Network) that supports distribution and streaming of such data as music, video and image. CDN consists of (1) contents servers that provide original contents, (2) server assignment server that assigns an appropriate contents server to each contents request from user side, and (3) high speed network that transfers contents. Since the contents servers are accessed by multiple users simultaneously, it must handle multiple media with keeping multiple sessions at the same time. At the user side, user appliance must handle multiple media at the same time although session is kept single. Server assignment servers must process many user requests simultaneously. Network in nature needs handle multiple sessions simultaneously. This situation causes the need of real-time multi-processing in the environment because multimedia must be handled simultaneously with controlling multiple communication sessions under specific time constraint of each media.
In case that eBusiness applications are realized on the application servers, three-tier model is generally adopted. The platform of the application servers are generally component-based by use of EJB (Enterprise Java Beans) and CORBA (Common Object Request Broker Architecture). Web servers and function servers are installed on top of the common application servers platform. Especially in the case of transaction processing, time-constraint of process is much more severe. Hence, the application servers platform must need real-time multi-processing in order to handle simultaneous processing of multiple processes with time-constraints.

Communication protocols in such layers as layer 2 (ATM, Ether), layer 3 (IP), layer 4 (TCP/UDP (User Datagram Protocol)) must simultaneously handle multiple sessions, connections, and datagrams, which cause the necessity of multi-processing capability.

Thus, real-time multi-processing is essential in the multimedia networking environment.

3.2 Ineffectiveness of Pseudo Multi-Processing in Sequential Processor

The author evaluated pseudo multi-processing of TCP/IP in sequential processors [7]. Two workstations accommodating CORBA, one of which is a client and the other is server, are interconnected through Ether network. The client generates threads and each thread sets up a TCP connection. On each TCP connection, the client requests process execution to the server. At every moment when a request arrives at the server from each client thread, server generates a thread. Several kind of processing time per thread were measured 10 times by changing number of threads from 1 to 45. Actually, Process execution time at the server and TCP/IP data transfer time on Ether were measured. The both times increased in proportion to thread counts as shown in Fig. 3(a). The increase in time per thread is apparently caused by thread switching overheads.

Besides, several approaches to realize TCP/IP communications boards, which shift the protocol handling to hardware on the boards from software on the workstations or PCs, were done to satisfy shorter time constraints. These boards achieve design throughput when IP datagram is longer than 4096 byte. Header handling is main process in TCP/IP handling. Therefore, process is lighter when IP datagram is long because rate of header relatively reduce. However, length of IP datagram will be about 200 byte in multimedia communication. For example, length of Transport Stream Packet (TSP) in MPEG2 is 188 byte.

Hence, processor architecture, which can realize real-time multi-processing without any runtime overheads, should be applied to protocol handling.

3.3 Experimental Study on Real-Time Multi-Processing

The CUE project examined data-driven implementation of protocol multi-processing to show efficiency of data-driven implementation. We firstly implemented TCP/IP on a CUE-p board [2]. In client server model, a client requires servers to communicate another client. Servers then process synchronizing and multicasting media such as video, sound, documents and control information. Besides, we implemented GIOP/IIOP in CORBA to realize interoperability on networking environment. In this study, we evaluated multi-processing capability in these protocol handling using the CUE-v1 processors [5]. We assumed bandwidth, data length and input rate as shown in Table 1. We consider MPEG4 as video, PCM as sound and typical Web page as a document. Considering actually maximum throughput of OC-3 ATM (135 Mbit/sec.), maximum number of users which need to process concurrently is 35.

Figure 3(b) shows evaluation result of real-time multi-processing in protocol handling. In all media, turn-around

<table>
<thead>
<tr>
<th>Media</th>
<th>Bandwidth</th>
<th>Data length (per datagram)</th>
<th>Input interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video</td>
<td>2 Mbit/sec</td>
<td>(250 byte)</td>
<td>1 msec</td>
</tr>
<tr>
<td>Sound</td>
<td>64 kbit/sec</td>
<td>(40 byte)</td>
<td>5 msec</td>
</tr>
<tr>
<td>Document</td>
<td>1.8 Mbit/sec</td>
<td>(1408 byte)</td>
<td>7 msec</td>
</tr>
<tr>
<td>Control Information</td>
<td>1 kbit/sec</td>
<td>(100 byte)</td>
<td>100 msec</td>
</tr>
</tbody>
</table>

Fig. 3 Real-time multi processing capability.
time can be kept constant when number of users increase by 35. These results demonstrate ideal multi-processing capability of data-driven protocol handling without any side-effects among multiple processes in progress concurrently. Furthermore, each turn-around time in movie, sound and control information is kept minimum value which equals to execution time for header management. In the other word, parallel implementation in the data-driven protocol handling can ensure fixed execution time when data length is shorter than 250 byte.

Thus, data-driven parallel realization of protocol handling demonstrated efficient real-time multi-processing capability without any runtime supervisory control as long as data length is appropriate as shown in Table 1.

4. A Networking-Oriented Data-Driven Processor and Future Directions

After experimental verifications of discussed Sect. 3, CUE project has been carried out to develop a networking-oriented data-driven processor which covers broader layers including applications in the networking environment. This section first describes current network processor architectures to show their typical design targets. Then, the latest version of our data-driven processor CUE-v2 is briefly introduced to show current status of CUE project. Effectiveness of CUE processor architecture as one of the most promising future VLSI realizations will be discussed.

4.1 Trends in Network Processor Architecture

Through advances in recent data transmission technology, the potential bandwidth of network links has been boosted exponentially. Accordingly, more value-added and complex networking applications such as VoD (Video on Demand) and media transcoding (e.g., to convert or compress a video stream from a high-speed link to a low-speed link at network junction points) have become possible to deploy in terms of bandwidth. To accommodate the next generation broadband multimedia networking, there appears to be a new breed of processor called network processor. Network processors are intensively optimized to exploit the inherent parallelism of packet processing. Although their detailed optimization methodologies differ considerably, it seems to establish the wide consensus for the extraction of TLP (Thread-Level Parallelism) due to the abundance of explicit packet-level parallelism in the network workloads [10]. Furthermore, network processors provide multithreading and low latency context-switching, since the processors hide the long latencies of memory references while analyzing the headers by swapping contexts with other threads. One typical example of the network processor architecture is chip multi-processor comprised of multiple cores supporting so-called hardware multithreading. Presently, most of network processors don’t exploit ILP (Instruction-Level Parallelism), since the packet processing done in current routers is simple and limited for header fields. In the advanced future network scenario, for example, real-time media transcoding becomes common, it is expected that the flexible exploitation of various levels of parallelism becomes more essential due to the abundance of ILP in media processing for payload fields of packets.

4.2 Data-Driven Chip Multi-Processor Core: CUE-v2

To create a streamlined architecture for future multimedia networking, we have studied a multimedia networking oriented data-driven processor. Data-driven architectures can naturally and efficiently exploit maximum and various levels of parallelism and spread it among processing elements and into their pipeline stages. Furthermore, data-driven architectures have benefits for architectural level real-time processing as was discussed in Sect. 3.3 because they offer constant execution time of each process and deterministic guarantees for packet rate due to their fair multiprocessing at instruction level without context switching overheads. This means that data-driven architectures can perform real-time processing without any runtime scheduling overheads such as real-time OSs assuming the deterministic input rate of a network node, which is, for example, wire rate.

Also, we have evaluated CORBA (Common Object Request Broker Architecture) protocol off-loading over OC-3 ATM (Asynchronous Transfer Mode) and real-time video compression using the CUE-v1 processors [5] built in a 0.25 \( \mu \)m CMOS process as shown in Fig. 1. These studies demonstrated that the CUEv1 achieved the real-time processing of actual applications for multimedia networking environment as we had expected.

These studies also gave us several issues to streamline the execution of the applications for multimedia networking. One of the issues is the inevitable inefficiency on serial codes, such as connection/port management in TCP and the serialization of parameters in video compression. Since data-driven architectures cannot exploit the locality of computation, it is not good at sequential processing. This is the compensation of exploiting fine-grained parallelism. To alleviate this issue with retaining the advantages of pure data-driven, we proposed an architecture which can simultaneously process data-driven and control-driven threads using common pipeline resource. The CUE-v2 performs both as data-driven and as out-of-order superscalar in order to alleviate the bottlenecks caused by sequential processing.

That is, a data-driven processor has benefits for real-time processing, such as instruction-level parallel processing without context switching overheads and fair resource scheduling among multiple threads. However, the following problems of data-driven architecture was pointed out in previous researches:

1. the potential inefficiency on serial codes,
2. the overhead of firing control, and
3. low packet distribution/input rate.

Today’s VLSI technology can solve the problem 2. and 3. by integrating a large CAM (Content Addressable Mem-
ory) and non-blocking switches, respectively. But the VLSI progress cannot solve the problem 1. This is because data-driven principle ignores the locality of computation. That is, given N as the pipeline depth of a circular pipeline of a pure data-driven processor and p as the parallelism in a program, CPI (Cycle Per Instruction) corresponds to max (1,N/p). Consequently, the instruction fetch rate or the frontend bandwidth of a data-driven processor is determined by the parallelism. Therefore the CPI of a serial code corresponds to N in data-driven architecture.

To address this problem, we proposed processor architecture for simultaneously processing data-driven and control-flow threads in a single pipeline. One kind, denoted “data-driven” is appropriate for the highly parallel parts; the other, “control-flow” is for the parts of the code with little parallelism. Unlike conventional data-driven/von Neumann hybrid architectures, exclusive execution of the control-flow thread is not performed in our proposed architecture. That is, the data-driven thread has priority to the control-flow thread in instruction fetching time. But, minimal issue opportunity is given to the control-flow thread to avoid blocking. Our architecture basically allocates empty slots, caused by the execution of the data-driven thread like in the failure of waiting-matching operation, to the control-flow thread. In addition, our architecture employs the forwarding paths for the control-flow thread for the speedup on the execution of dependent instructions. To do this, we introduce the extended the firing control unit, which also functions as a reservation station in out-of-order superscalar processors.

Figure 4 shows the pipeline structure of the CUE-v2. The CUE-v2 seems similar to a superscalar machine except its circulation path for data-driven thread and its thread management units. In fact, control-driven thread is processed like as a nonspeculative 2-issue out-of-order superscalar machine does. In view of a traditional dataflow machine, the CUE-v2 adds a program counter and architectural register to front-end pipes. To share between both type threads as much hardware resource as possible, the modules used by only one side of thread type are minimized.

The functionality of each pipeline stage is stated as follows: Instruction Fetch 0(IF0) stage selects a thread type to be fetched and calculates instruction addresses. This calculated addresses are forwarded to Instruction Fetch1(IF1) stage. IF1 stage fetches instructions and issues to Instruction Decode0(ID0) stage. ID0 stage decodes instructions and reserves a matching memory field at Firing Control (FC). Here, we represent the address of this field as MMA (Matching Memory Address). Note that FC stage and Write Back (WB) stage release the MMA in data-driven thread and in control-driven thread, respectively. ID1 stage dispatches instructions to the FC according to the MMA. In addition, the ID1 stage accesses a register file and performs register renaming in case of control-driven thread. FC stage is responsible for out-of-order scheduling. That is, incoming instructions in both type of thread wait until their source operands become available in FC stage. In addition, it waits until the directions of previously issued branch instructions are determined in case of control-driven thread. INTOegero,1/Load-Store (INT0,1/LS) stage executes an incoming instruction in both type of thread. In case of data-driven thread, it transfers to SWitch (SW) stage for circulating a data-driven packet after calculating a next instruction address; in case of control-driven thread, it transfers to write back path and broadcasts a result operand to matching memories in FC stage. Branch (BR) executes control-driven branch instructions and sends a branch direction to branch predictors and FC stage. The CUE-v2 equips BTAC (Branch Target Address Buffer) and BHT (Branch History Table) for branch prediction. If miss branch prediction occurs, BR stage issues a recovery request to front-end pipes, FC stage, and snapshot to eliminate whole in-flight state instructions in pipelines.

The prototype chip was developed by employing standard-cell design, and it was implemented using timing-driven synthesis/layout. Crossstalk, antenna effect, and voltage drop were analyzed and validated using commercial EDA tools. The chip is built in a generic 0.18µm six-metal layer process, the die size of 5 × 5 mm², including 64 kbyte SRAM, and is packaged in a ball grid array having 292 pins. The chip layout is shown in Fig. 5(a). The chip is verified with several applications. It proves to work without any significant flaws in this chip. Most of its functional verification time was spent for an out-of-order scheduling of control-flow. The verification for basic operations and the simultaneous processing of dataflow and control-flow threads was not dominant. Thus, the CUE-v2 architecture is not more complex than superscalar processors. The CUE-v2 chip will be installed to PCI board in Fig. 5(b), which was originally designed in the CUE project, to be examined for its potential capability as chip multi-processor core in the next generation CUE-v3.

4.3 Toward VLSI-Oriented Architecture in the Sub-100 nm Era

Power consumption is an essential issue for integrating more than a hundred million transistors on a single chip [11].
In order to reduce power consumption, various researches have been carried out on every implementation levels such as semiconductor processes, circuit designs and architectures. Needless to say, any improvements in the semiconductor processes and circuit designs are always effective to reduce power consumption regardless of which architecture is adopted. In this section, an architectural method as a personal prospect is discussed to achieve lower power consumption from a viewpoint of a researcher who has been studying on-chip multi-processor realizations of data-driven architecture.

One of the most crucial issues in realizing lower power architecture is to establish efficient parallel processing scheme without any runtime overheads. It may sound strange but still true that this requirement becomes clearer by carefully taking account of development history of parallel processing systems. Apparently, every runtime overhead results in consuming redundant power. As was well-known, most of parallel processing systems have been developed by interconnecting conventional sequential processors. Additional runtime control in these trials resulted in showing difficulties for both software and hardware in parallel processing system. The experimental results for C.MMP and its operating system HYDRA [12], which were world-famous as one of the earliest prototype, were fine examples. Additional runtime overheads in parallelizing system still remain in most of parallel processing systems.

Considering these facts, the author has been studying data-driven principle as the most natural scheme as long as representation and execution of parallel processing are concerned. One of the most fruitful results is the real-time multi-processing scheme without any runtime control in the data-driven parallel implementation as described in Sect. 3. Furthermore, the author has been striving for the most efficient VLSI realization of the stream processing based on the dynamic data-driven execution scheme. As was emphasized in Sect. 2, one of the most salient features of our data-driven processors is that the “flow-thru” processing scheme is extensively employed throughout the system to form an integrated elastic pipeline system. Although pipeline processing scheme itself is popular in many parallel processing systems, it has a particular significance in a data-driven execution since a pipeline stage can show its ultimate effectiveness with extremely simple self-timed control. These are due to fundamental nature of data-driven execution where all the fired processes can be executed independent of any other processes being fired concurrently. This means that every processes in real-time multi-processing can be organized just to do simple unidirectional or straight-forward pipeline processing without any runtime overheads such as interlocking or flushing with other processes in progress concurrently.

The “flow-thru” processing scheme adopted in our data-driven processors was demonstrated to exhibit ideal improvement of performance according to progress of fabri-
cations as shown in Fig. 6. Note also that our VLSI realization is highly effective as a lower power architecture since power consumption is limited to the only active stages.

Figure 7 shows typical turn-around time and throughput in a circular pipeline adopted as a chip multi-processor core in the CUE-series data-driven chip multi-processor core. As shown in Fig. 7(a), the turn-around time keeps minimum value as far as over loaded condition is avoided. This nature of the pipeline fully utilized in the real-time multiprocessing discussed in Sect. 3.3. Also, it was experimentally verified that power consumption of our data-driven processors was in proportion to throughput achieved in the pipeline as shown in Fig. 7(b). This feature and passive operation mode in ready-to-fire principle of the data-driven scheme will be effective to minimize power consumption at standby time.

5. Conclusion

This paper proposed and discussed a novel design philosophy to achieve both sufficient throughput and efficient real-time multiprocessing essentially needed in multi-media networking environment.

Considering expandability and lower power consumption, the elastic circular pipeline developed in the CUE project is one of the most promising cores in the chip multi-processor architecture for fully utilizing future VLSI progress.

The author strongly feels that sub-100 nm era is an appropriate time to shift from conventional machine first approaches to user- or demand-centered paradigm.

Although the networking-oriented data-driven chip multi-processor CUE is still the first stage of its realization, the CUE project promises to provide outstanding platform to build a secure networking infrastructure.

Acknowledgement

Although it is impossible to give credit individually to all those who organized and supported the CUE project, the author would like to express his sincere appreciation to all the colleagues in the project.

This research is partially supported by the Scientific Grant of Japan Society for the Promotion of Science and Semiconductor Technology Academic Research Center.

Finally, the author would like to express his sincere thanks to Emeritus Prof. Hiroaki Terada of Osaka University for his invaluable suggestions in novel parallel processing architecture.

References

Hiroaki Nishikawa received B.E., M.E. and Ph.D. degrees in electronic engineering from Osaka University in 1976, 1981 and 1984, respectively. After being with Osaka University, he is presently a Professor in Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba. He was a visiting scientist at Laboratory for Computer Science, MIT from July 1994 to August 1995, from November to December 1997 and from April to May 1998. He also worked as a visiting professor in the Electrical Engineering and Computer Science Department at the University of Southern California from March to April 1988. His current research interests include user-friendly system specification and verification environment, VLSI-oriented data-driven processor architecture and hyper-distributed system. He received IASTED Best Paper Award in the area of Processor Architecture in PDCS 2003. Dr. Nishikawa is a member of IPSJ and a senior member of IEEE.